Poročilo:

O izvedeni vaji:

19\_lav03\_vaja

Laboratorijske vaje

Vaja 19

Maj Mohar, G3A

Ljubljana, marec 2025

Naloga 1\*

Objekt vrste Smer predstavlja trenutno smer gibanja lika, s katerim je povezan. Gibanje je ravninsko in zgolj ortogonalno /recimo vzporedno z osmi koordinatnega sistema ravnine/.

Vektor (normaliziran) gibanja je/sta privatni lastnosti objekta in lahko zavzema samo 4 vrednosti, za vsako od smeri po ena. Tipično objekt vrste Smer inicializiramo z smerjo gibanja, objektu, ki ga ne inicializiramo s smerjo določimo privzeto smer gibanja 'desno'. Za vsako komponento vektorja gibanja mora obstajati vračalec/getter/. Zagotovite možnost kontrolenega izpisa objekta na zaslon. Objekt vsebuje metodi obrniLevo/0 in obrniDesno/0, ki popravi smer tako, da v prvem primeru 'zasuka' smer za 90 stopinj v smeri nasprotni urinemu kazalcu, in druge za 90 stopinj v smeri urinega

kazalca.

a) iz opisa konstruirajte ustrezen razred, kreirajte par objektov te vrste in preskusite:

- omejitve pri konstrukciji objekta

- zmožnost spreminjanja smeri v skladu z opisom

/ preskus in dokazovanje delovanja sta v vaši domeni /

b) izrišite razredni diagram za razred Smer

Naloga 2\*

Lik v tej nalogi bo dokaj abstraktna zadeva. Ni opredeljen z obliko, temveč zgolj s pozicijo v ravnini in smerjo gibanja,ki jo določa vsebovan objekt vrste Smer. Seveda so vse lastnosti vsakega Lik-a privatne. Lik kreiramo tako, da mu podamo pozicijo v ravnini in smer gibanja. V primeru, da smeri ne podamo, se uporabi privzeta smer. Če Lik kreiramo brez argumentov, se postavi v izhodišče koordinatnega sistema s privzeto smerjo gibanja.Javna metoda premakni/0 premakne objekt za 1 v trenutno z objektom določeni smeri gibanja. Javni metodi obrniLevo/0 in obrniDesno/0 lik obrneta, kot je opisano v Smer. Za pozicijo lika mora/ta v objektu obstajati ustrezna vračalca.

a) iz opisa konstruirajte ustrezen razred, kreirajte par objektov te vrste in preskusite:

- premikanje

- zmožnost spreminjanja smeri pri gibanju lika

b) izrišite razredni diagram / nivo podrobnosti: omejite vsebina na zgolj javne člane

\* Uporabimo implementacijo iz realizacije vaje s številko 18

Naloga 3

Lik specializiramo v krožnico (Krogec) s tem da dodamo lastnosti, ki opredelijo krožnico :

središče(ravninska koordinata), radij (neceloštevilska vrednost). Npr.

class Krogec extends Lik {

int centerX, centerY;

double radij;

}

a) preskusite, če lahko Krogec naredite s 'privzetim konstruktorjem', kot npr.: Krogec k = new Krogec();

Če se da, izpišite vrednosti VSEH lastnosti iz objekta 'k' in preverite, če deluje enako, kot konstruktor brez argumentov baznega razreda (Lik).

b) dodajte v definicijo Krogec kontruktor : Krogec(){super();} . Ponovite zahteve pod alinejo a). Kaj opazite.

Opazim, da se uporabi konstruktor iz prvega razreda Lik in ne od Krogec

c) dodajte lasten konstruktor (preobložitev/overload), ki bo ob kreiranju objekta vrste Krogec tega inicializiral s središčem, smerjo gibanja in radijem. Prijemališče lika/krožnice naj bo podano središče. Prijemališče in smer inicializirajte s klicem konstruktorja baznega razreda (Lik-a).

d) preverite, ali delujejo funkcije obračanja smeri podedovane iz Lik

Ja, delujejo.

e) preverite, ali deluje funkcija izpisa Krogca, kadar ga želite izpisati kot : System.out.println(k);

Izpiše vrsto objekta in pa prostor, kjer se nahaja

Naloga 4

Metoda toString v Krogec ni ustrezna. Radija ne dobite vizualiziranega. Popravite mehanizem tako, da se bo izpisal še radij krožnice. (prekrivanje/overrun metode)

public String toString(){

**return** ("Center kroga: T("+centerX+", "+centerY+"), radij: "+radij+", smer: S("+super.s.getX()+", "+super.s.getY()+").");

}

Naloga 5

V definicijo Krogec dodajte metodi, ki bosta izračunali in vrnili obseg ter ploščino krožnice na osnovi radija zapisanega v objektu.

public double izracunajObseg(){

**return** (Math.PI \* 2 \* radij);

}

public double izracunajPloscino(){

**return** (Math.PI \* radij \* radij);

}

Naloga 6

Naredite še specializacijo Lik v Trikotnik. Ta je opredeljen s koordinatami svojih oglišč. Eno izmed oglišč je tudi vedno prijemališče tega lika. Ustrezno opredelite potrebne konstruktorje in pazite, da se bo trikotnik ob premiku tudi ustrezno premaknil. Vsa ostala funkcionalnost naj bo enakovredna tisti pri Krogec, ključno z izračunom ploščine in obsega.

Koda je v prilogi in na koncu dokumenta.

Naloga 7

Ponovite 3. in 4. nalogo 18.vaje. Pri realizaciji kolekcije naredite 2 tabeli; ločeno za Trikotnike, Krogce.

če se vam da : Uporabite tabelo likov kot je v vaja18/3, nato v tabelo dodajate naključno trikotnik ali krogec. Pri premikanju, vizualizaciji, ko imate dostop do elementa tabele, glede na to, ali je v tabeli vpisan trikotnik ali krogec ( t[i] instanceof Trikotnik) izvedite vsiljeno pretvorbo v ta tip in šele nato npr vizualizacijo objekta ...

**import** java.lang.\*;

**import** java.util.\*;

/\*\*

\* Opis: Razred Trikotnik je razširjen razred Lik, specializiran za lastnosti in metode za trikotnik

\*

\* **@author** Maj

\* **@version** 31. 3. 2025

\*/

class Trikotnik extends Lik {

int x1, y1, x2, y2, x3, y3;

/\*\*

\* Konstruktor brez argumentov poda privzete točke

\*/

public Trikotnik() {

super(0, 0);

this.x1 = 0;

this.y1 = 0;

this.x2 = 1;

this.y2 = 0;

this.x3 = 0;

this.y3 = 1;

}

/\*\*

\* Kostruktor prejme koordinate trikotnika in jih zapiše

\*/

public Trikotnik(int x1, int y1, int x2, int y2, int x3, int y3) {

super(x1, y1);

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.x3 = x3;

this.y3 = y3;

}

/\*\*

\* Funkcija kliče premakni, ki premakne prvo točko, vse ostale pa premakne enako kot je prvo

\*/

public void premakni() {

super.premakni();

int dx = super.getX() - x1;

int dy = super.getY() - y1;

x1 += dx; y1 += dy;

x2 += dx; y2 += dy;

x3 += dx; y3 += dy;

}

/\*\*

\* Fukcija izračuna obseg trikotnika

\*/

public double izracunajObseg() {

double a = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));

double b = Math.sqrt(Math.pow(x3 - x2, 2) + Math.pow(y3 - y2, 2));

double c = Math.sqrt(Math.pow(x1 - x3, 2) + Math.pow(y1 - y3, 2));

**return** a + b + c;

}

/\*\*

\* Fukcija izračuna ploščino trikotnika

\*/

public double izracunajPloscino() {

**return** Math.abs((x1\*(y2 - y3) + x2\*(y3 - y1) + x3\*(y1 - y2)) / 2.0);

}

/\*\*

\* Funkcija preobloži metodo toString in uporabi svojo

\*/

public String toString() {

**return** "Trikotnik s točkami: A(" +x1+ ", " +y1+ "), B(" +x2+ ", " +y2+ "), C(" +x3+ ", " +y3+ "), smer: S(" +super.s.getX()+ ", " +super.s.getY() + ").";

}

}

**import** java.lang.\*;

**import** java.util.\*;

/\*\*

\* Opis: Razred Krogec deduje vse lastnosti iz razreda Lik in se specializira za krog

\*

\* **@author** Maj

\* **@version** 17. 3. 2025

\*/

class Krogec extends Lik {

int centerX, centerY;

double radij;

public Krogec(){

super();

centerX=super.getX();

centerY=super.getY();

radij=1;

}

public Krogec(int x, int y, double radij){

super();

this.centerX = centerX;

this.centerY = centerY;

this.radij = radij;

}

public Krogec(int centerX, int centerY, double radij, int s1, int s2){

super(centerX, centerY, s1, s2);

this.centerX=centerX;

this.centerY=centerY;

this.radij = radij;

}

public String toString(){

**return** ("Center kroga: T("+centerX+", "+centerY+"), radij: "+radij+", smer: S("+super.s.getX()+", "+super.s.getY()+").");

}

public void premakni(){

super.premakni();

}

public void obrniLevo(){

super.obrniLevo();

}

public void obrniDesno(){

super.obrniDesno();

}

public double izracunajObseg(){

**return** (Math.PI \* 2 \* radij);

}

public double izracunajPloscino(){

**return** (Math.PI \* radij \* radij);

}

}